DEVELOPMENT OF MARCH TEST GENERATOR USING C++

TAN EWE CHEONG

UNIVERSITI TEKNOLOGI MALAYSIA

DEVELOPMENT OF MARCH TEST GENERATOR USING C++

TAN EWE CHEONG

A project report submitted in partial fulfillment of the requirements for the degree of Master of Engineering (Electrical – Microelectronics and Computer System)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

NOVEMBER 2009

ABSTRACT

Semiconductor memories are widely considered as one of the most important types of microelectronic components in the modern digital systems. The growing need for storage in computer, communications and consumer applications is driving the continuous innovation of various semiconductor memory technologies. Memories are more vulnerable to physical defects than logic circuits because of the former have high density and more complicated processing steps. Hence, investing in fault modelling and simulation, test algorithm development and evaluation, design-for-testability (DFT), built-in self-test (BIST) and diagnostics has been considered as one of the key factors in producing successful memory as well as system-on-chip (SoC) products. Tools for fault model evaluation and test algorithm generation are fundamental for tackling the above issues efficiently. March test algorithms are developed to detect these faults. Majority of the published march tests have been manually generated. Unfortunately, the continuous evolution of the memory technology introduces new classes of faults. This makes the task of handwriting test algorithms harder and as well the task of analyzing these algorithms becomes more complicated. Doing this manually does not always lead to optimal results. Therefore, it is becoming evident that it is more feasible to perform algorithm generation and fault simulation in an automated way. In this thesis, we develop a program to generate March test automatically. The focus is to ensure that the generator is able to product March tests that cover most typical faults in memories. Each March test generated is carefully analysed to ensure that it meets the required fault coverage. Prior to this, the characteristics of each memory fault are studied so that fault behaviour of the particular is correctly represented in form of fault primitive. As a result, the generator developed in this project is able to correctly generate March test with full coverage of the target fault.

ABSTRAK

Memori semikonduktor merupakan salah-satu komponen mikroelektronik penting dalam sistem digital moden. Permintaan terhadap saiz memori yang kian meningkat dalam aplikasi komputer and komunikasi, telah menjadi satu pendorong utama terhadap perkembangan teknologi memori semikonduktor. Memori lebih cenderung terjejas akibat kerosakan fizikal berbanding litar logik. Ini disebabkan memori mempunyai ketumpatan sel memori yang lebih tinggi dan memori mempunyai langkah-langkah pemprosesan yang jauh lebih rumit. Oleh itu, dalam menghasilkan produk memori yang berkualiti, tumpuan harus diberikan kepada analisis kegagalan memori, model sesaran dan simulasi, penyelidikan dan penilaian ujian algoritma dan lain-lain. Peralatan untuk penilaian model sesaran dan generasi ujian algoritma adalah asas kepada penyelesaian efisyen untuk isu-isu tersebut. Ujian algoritma March adalah penting dalam mengesan sesaran dalam memori. Kebanyakan ujian March yang telah diperkenalkan, sebenarnya dihasilkan secara manual. Malangnya, dengan evolusi berterusan dalam teknologi memori, terwujudlah kelas-kelas sesar yang baru. Ini menyebabkan tugas-tugas generasi ujian algoritma secara manual atau tulisan tangan, menjadi lebih rumit and sukar. Justeru itu, adalah lebih munasabah jika ujian algoritma March dihasilkan secara automatik. Dalam projek ini, satu pendekatan untuk menghasilkan ujian March secara automatik akan dilaksanakan. Objektif utama adalah untuk memastikan penjana ujian March dapat menghasilkan ujian March untuk sesaran-sesaran memori yang lazimnya berlaku. Setiap ujian March dianalisa secara teliti untuk memastikan ia memenuhi keperluan liputan sesaran. Sebelum itu, ciri-ciri setiap sesarn memori dikaji dengan teliti. Ini adalah supaya ciri-ciri memori sesaran dapat diwakili dalam bentuk sesaran primitif. Oleh itu, pejana ujian memori ini dapat menghasilkan ujian march dengan liputan sesaran yang penuh.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This project involves the development of a memory March test generator using C++. The generator outputs a March test which provides full coverage of the target fault specified by the user. This chapter covers the background, problem statement, objectives, scope of work, methodology, contribution of this work and finally the thesis organization.

1.1 Background

Memories are one of the most important components in digital systems and semiconductor memories are nowadays one of the fastest growing technologies. Over the past thirty years, memory has changed from a few high-volume stand-alone standardized MOS memory to embedded memories in CMOS logic processes to potential universal memory technology in numerous instances in multimillion transistor logic chips. It is forecasted in (Marinissen et. al, 2005) that embedded memories will reach 90% of the chips' area surface in 10 years. The recent popularity of advanced technologies in computer, communications, consumer and network applications, has boosted the need for storage in these fields. Bigger and faster memories are always desirable for voluminous transmission and storage data in these applications. Even today's system-on-chip (SoC) is moving from logic-dominant chips to memory-dominant chips. With semiconductor memory being the most dense physical structure, the increasing size and density of memory chips will soon make their testing the bottleneck of the entire production process. Besides,

memories are designed with aggressive design rules especially for embedded memories. Consequently, they tend to be more prone to manufacturing defects and reliability problems. This in turn impacts the production yield as it is largely dependent on memories and the development of efficient test solutions. Thus, securing high memory yield is critical to achieving lower-cost silicon. So it is without doubt that testing semiconductor memories has become increasingly important.

1.2 Problem Statement

In recent years, the capability of integrating hundreds of millions of gates and cells on the chip means that large subsystem sections are being integrated and memory is becoming a large part of a chip which is functionally not a memory. However, despite this, the common issues in memory testing are still fundamentally there. The issues in memory testing are, namely, the characterization of detailed and realistic fault models and the definition of time-efficient test algorithms able to detect them.

Faults in memories are usually modelled as functional faults so that functional test can be used to detect those faults. To test memories, the types of memory faults have to be identified. It is common that functional fault models are used for memories. They define the functional behaviour of the faulty memory. The fault models usually do not require knowledge of the memory chip at circuit level. A higher level of abstraction is used. Memories are represented by a set of blocks hiding implementation details and signal levels are represented by logical values. However, the issue here lies in the difficulty of precise modelling of the behaviour of the memory. As we move to very deep-submicron process technologies, this task has become more and more challenging. There might be new classes of fault that emerge which appear to be more and more problematic from the test point of view. The challenge here would be for fault model to be correctly defined in order to be able to catch the new fault. In the previous years, static faults (faults sensitized by the

execution of a single memory operation) have been the predominant class of fault models addressed by researchers and the industry. However, of late, the emergence of new faulty behaviours like dynamic and linked faults has brought to new classes of fault models.

Over the years, many different types of algorithms were proposed and yet March tests have been proven to be simpler, faster and regular structured, among the different types of algorithms that were proposed. March tests are generated based on the fault models. A good test algorithm is an algorithm that has complete coverage of the target faults and is also time-efficient, that is it consumes lesser test time when executed. Lesser test time translates to lower cost of testing. The cost of testing memories increases rapidly with the new memory chips with deeper submicron technology (Inoue *et al.,* 1997). Hence, in order to keep cost and test time within economically acceptable limits, it is essential to have precise fault modelling and efficient test design. The quality of the used tests, in terms of their fault coverage and test length, is strongly dependent on the used fault models. Hence, this justifies the need to have a time-efficient test algorithm. Similar to the first issue, the challenge is greater as the technology gets more complicated as the test algorithm requires more test time. Basically, to define a test algorithm, one has to juggle between test time and fault coverage.

As mentioned earlier, a lot of studies have been done on static faults as such many march tests have been proposed to cover this class of fault. These March tests were manually generated. Analysis of the efficiency of each of them was manually done. A lot of efforts were required in defining the test algorithms for static faults. Besides, doing this manually doesn't always lead optimal result. As a whole, such manual definition can be pretty time consuming. This will not be any easier for new classes of faults which certainly would be more complicated. An example would be the amount of effort needed to define test algorithms that detect dynamic faults. This brings to the need to have an automated way to define and produce the optimal March test algorithm. This would eliminate the tedious work of generating the March algorithm manually or by hand. Having this would certainly save a lot of time for the test designers. Hence, this addresses the mentioned problem statements and brings to

the work of this project. In this project, a fault formalism to model the behaviour of the fault is introduced and an automated way of generating March test algorithm is developed.

1.3 Project Objectives

From the problem statement in the previous section, two objectives have been set out for this work, which are:

1. To develop a tool that generates March test algorithm automatically based on the list of target faults provided by user.
2. To generate test algorithm that have full coverage on the specified target faults and is able to cover common faults in memory.

1.4 Project Scope

The following are the scope of this project:

1. This project involves the development of memory algorithm generator. The generator is developed using C++ language, Microsoft Visual C++ 2008.

2. The March algorithm is generated based on an input from a list of target fault list provided by user. Fault list should contain the list of specific faults to detect. The algorithm for the memory test algorithm generator will be based on the approach presented by Benso *et al.* (2008). It is a new approach to automatically generate March test targeting static and dynamic faults. This approach has introduced a completely new March test generation algorithm with a polynomial complexity that strongly reduces the generation time.

3. The generated test algorithm will need to be analysed to ensure that all target faults have been covered or can be detected by the algorithm.

4. The project also includes the fault model and fault formalism in the form of fault primitives for common memory faults.

Figure 1.1 shows the free body diagram of the generator. Basically the generator consists of two main sub-blocks which are the fault input converter and the March test generator. The fault input converter translates the target fault list provided by the user to a proper form known as test pattern. In turn, the test patterns are fed to the March generator which generates out the test algorithm that covers the faults specified in the target fault list.

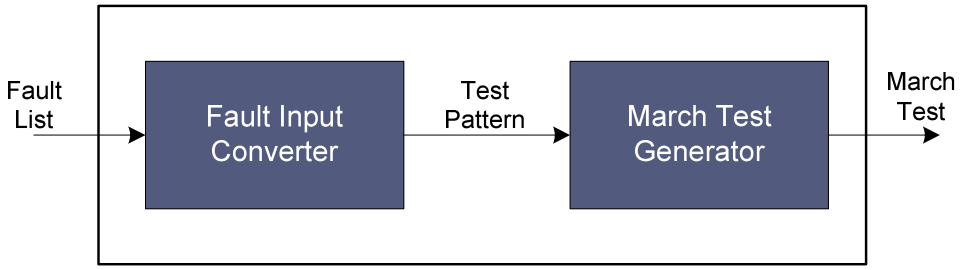

Figure 1.1: An Overview of the March Generator

1.5 Project Methodology

1. First, I have done literature review on memory testing, specifically the previous works of March test generation. A numerous papers were read on test algorithms and memory faults. Some are discussed in the "Literature Review" section. Besides that, books were also read to have a better understanding on this topic that is the memory testing.

2. Upon understanding the scope of interest, the possible improvement of March test generation were studied and identified. Notable one would be the fault list converter, details would be explained later.

3. This is then followed by the derivation of finite state machine based fault models for typical memory faults, based on a standard notation used in the March generation algorithm. This is very useful for the March test generation.

4. Then, March generation algorithm were read and understood. Pseudo-codes were constructed based on the understanding of the algorithm. This is then followed by implementing the pseudo-codes using C++, Microsoft Visual C++ 2008.

5. This stage was then followed by the analysis and verification of the generated test algorithm. Each March test generator is analysed carefully to ensure that all faulty edges are being covered. This is important to ensure that the generated test algorithm provides the expected fault coverage.

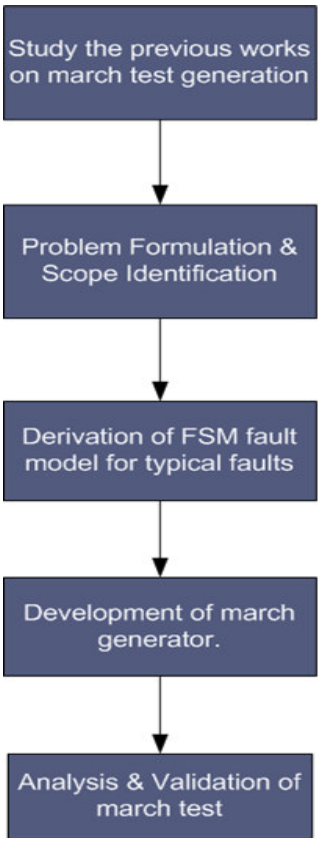

Figure 1.2: Summary of the project methodology

1.6 Work Contribution and Project Delivery

1. An automated March test generator is developed. It generates test algorithms that cover common faults in memory, namely static and dynamic faults.

2. The automated generator will help to reduce memory test development and evaluating effort significantly. It is flexible and can be generalized to cover other memory types and fault models easily.

3. Fault models of memory faults that provide better understanding of memory faults.

1.7 Project Organization

The thesis is organized into six chapters. The first chapter introduces the motivation, objectives, scope, methodology, contribution and together with project organization.

Chapter 2 reviews the background of the research. Related works similar to this field are presented. Summary of the literature review is given to clarify the rationale of this work.

Chapter 3 describes the memory fault models, types of memory faults and also the finite-state-machine based fault models for each memory faults mentioned.

Chapter 4 introduces the March test generation concept which is will be developed into a generator based on C++. The pseudo-codes of the automatic generation are also shown here.

Chapter 5 reports the result of the test algorithm generation. This also includes the analysis and verification work on each generated test algorithm.

In the final chapter, the project work is summarized and the potential future works are proposed.

REFERENCES

Marinissen, E.J., Prince, B., Keltel-Schulz, D., and Zorian Y.(2005). Challenges in Embedded Memory Design and Test. *Design Automation and Test in Europe*. Munich, Germany. Vol. 2 pp.722-727.

Inoue M., Yamada T. and Fujiwara A. (1993). A New Testing Acceleration Chip for Low-Cost Memory Tests. *IEEE Design and Test of Computers*. Vol. 10, no. 1, pp. 15-19,

Benso A., Bosio A., Di Carlo S., Di Natale G. and Prinetto P. (2008). March Test Generation Revealed. *IEEE Transactions on Computers*, vol. 57, no. 12.

Van de Goor, A.J., and Al-Ars, Z.(2000). Functional Fault Models: A Formal Notation and Taxonomy. *In Proc. of IEEE VLSI Test Symposium* pp.281-289.

Van de Goor, A.J. and Verruijt C.A (1990). An Overview of Deterministic Functional RAM Chip Testing. *ACM Computing Surveys*. Vol.22, No.1, 1990, pp.5-33.

Hamdioui S., Wadsworth R., Reyes J.D. and Van de Goor A.J. (2003). Importance of Dynamic Faults for New SRAM Technologies. *Eighth IEEE European Test Workshop (ETW'03)*.

A.J van de Goor (2004). Using March Test to Test SRAMs. *IEEE Design and Test Computers*, vol. 10, no. 1, pp. 8-14.

Hamdioui S., Al-Ars Z. and Ad J. van de Goor (2002). Testing Static and Dynamic Faults in Random Access Memories. *20th IEEE VLSI Test Symposium*. pp. 95-100.

Smit B. and A.J. van de Goor (1994). The Automatic Generation of March Tests, *Proc. IEEE Int'l Workshop Memory Technology, Design and Testing (MTDT'94)*, pp. 86-91.

Zarrineh K., Upadhyaya S. and Chakravarty S (2001). Automatic Generation and Compaction of March Test for Memory Arrays. *IEEE Transactions VLSI Systems*. vol.0, no. 6, pp. 845-857.

C.-F. Wu, C.-T. Huang, K.-L Cheng, C.-W. Wu. (2002). Fault Simulation and Test Algorithm Generation for Random Access Memories. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. vol. 21, no. 4, pp. 480-490.

Di Carlo S., Di Natale G., Benso A., and Prinetto P. (2002). An Optimal Algorithm for the Automatic Generation of March Tests. *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE '02)*. pp. 938-939.